

CTSS Reverse Chronological Table of Contents

AD.4		Common & Public	8/6/65
AH.1.01	LOGIN	Log in	7/1/65
AH.5.01	LISTF	List UFD	8/6/65
AH.1.04	TTPFFK	Query time and records	8/6/65
AG.2.02	SFFK, READK etc.	Input, buffered	7/30/65
AG.2.03	ASSIGN, WRITE	Output, buffered	7/30/65
AC.3		DATADUMPS	7/30/65
AH.9.05	MAIL	Send messages	7/30/65
AD.2		New file system	7/15/65
AG.2.08	OPEN, BUFFER etc.	I/O in new system	7/15/65
AG.3.07	CHFILE, DEFILE, ESTATE	Change mode and name	7/15/65
AG.5.05	MOUNT, UMOUNT etc.	Tapes in foreground	7/15/65
AG.7.03	UPDMFD, ALLOT, etc.	Privileged I/O calls	7/15/65
AH.3.06	MOUNT, UMOUNT, etc.	Tape commands	7/15/65
AE.1		Delayed bulk I/O	7/09/65
AG.3.01	CHMODE, RENAME	Change mode and name	7/09/65
AG.3.02	DELETE, ERASE	Delete and erase files	7/09/65
AH.6.03	CHMODE, RENAME, DELETE	Change commands	7/09/65
AH.6.06	RQUEST	Off-line processing	7/09/65
AH.7.01	LOAD etc.	Relocatable program	7/01/65
AG.2.06	SRCH, BLK, CLOUT, etc.	Library service routines	6/25/65
AG.2.10	BFOPEN, BFREAD etc.	Buffered I/O	6/21/65
AG.2.11		Write-arounds	6/21/65
AG.6.08	TILOCK	File-wait	6/21/65

CRIG 5x2

Aug 6 1965

Summary of TYPSET requests.

<u>abbrev.</u>	<u>request</u>	<u>response</u>
Basic requests:		
L	LOCATE string	line found * end-of-file
D	DELETE n	end-of-file
N	NEXT n	end-of-file
I	INSEPT line	none
P	PRINT n	printed lines, end-of-file
C	CHANGE 0xx0yy0 n G	changed lines *
T	TOP	none
	FILE name	Ready message
Special-purpose requests:		
B	BOTTOM	"Input" *
V	VERIFY ON (or OFF)	none
S	SPLIT name	no name given
R	RETYPE new line	none
F	EPASE x	none
K	KILL x	none
F	FIND string	line found * end-of-file

* These responses will not occur if VERIFY mode is off.

Revised: 1/3/66

Identification

CTSS Character Set

Purpose

Two character sets, one a subset of the other, are standard on CTSS. The smaller set (the 6-bit or BCD set) is basically the 7094 standard BCD set of 6-bit character codes including 47 characters and blank, and augmented with four console control functions. (Carriage return, tabulate, form feed, and colon, which is used by some programs as a logical "backspace" character.) The larger set (the 12-bit or Full set) consists of 111 graphic and control characters, represented as 7-bit codes right-adjusted in a 12-bit field. This larger set includes both upper and lower case letters and a variety of special characters and console control functions.

Twelve-to-six bit mapping

All input from consoles is treated initially as 12-bit codes by the CTSS supervisor. These 12-bit codes will, however, normally be mapped into the six-bit subset by the supervisor unless special action is taken by the user program to prevent the mapping. Supervisor calls (SETBCD and SETFUL) are available for turning on and off the mapping.

In the CTSS Character Set table below, the 6-bit subset is contained in the upper half of the table. When a character from the lower half of the table appears in an input stream, it is mapped according to the following rules:

1. Characters in the table in parentheses are discarded.
2. All other characters except double quote and question mark are truncated to six bits by discarding the left six bits.
3. Double quote is the "erase" character: the previous character is discarded.
4. Question mark is the "kill" character: the entire line is discarded.

To simplify the job of a program which wishes to do its own 12-to-6 bit mapping, the supervisor on input inserts a flag bit (the fourth from the left) on those codes which are to be discarded upon mapping. For example, the 12-bit code for the percent sign, according to the table, is:

000001000101

(0105 octal)

When using the RDFLXA supervisor call, the code which will be received by a user program will be:

CBI 61 box 2

Jan 3, 1966

000101000101

(0505 octal)

since this character is discarded when mapping to six-bit mode. The flag bit is optional on output characters. For example, to type out a percent sign, either code 0105 or 0505 is acceptable.

Device Code Tables

No one device is capable of input or output of the complete CTSS character set. For each device, a table is provided which lists the exceptions. In most cases, these tables indicate one of two mapping rules for exceptional characters. These rules are:

1. The character is discarded on output, or
2. The character prints as some graphic different from standard.

The fact that a different graphic is attached to a given code does not of course, imply that the code will be interpreted differently by the computer. This latter comment must be kept in mind when using a 1050 console, which may have any of several slightly different sets of key caps and/or printing balls.

On the Model 35 Teletype and the Telex, the upper and lower case letters are mapped together as in the following example:

1. On input, a typed letter "A" will always produce the code for upper case "A", 0021.
2. On output, the code for lower case "A", 0121, will type an upper case A.

Character Code Tables

Unassigned positions in the CTSS character set table are reserved for future expansion. At present, these unassigned characters are discarded on output. In the individual device code tables, a lack of an entry implies that the corresponding entry in the CTSS character set table applies. The entry "ig" means that this character code is ignored on output to this device.

All codes are given in octal.

Abbreviations used in the character set tables:

ig - Ignored (see comment above)
WRU - Who are you
P-off - Printer off
P-on - Printer on
V.T. - Vertical tab
C.R. - Carriage return
L.F. - Line feed
F.F. - Form feed
tab - Horizontal tabulation
hang - data phone disconnect
sngl - Single space carriage on return
dbl - Double space carriage on return
L.K. - Lock keyboard
U.K. - Unlock keyboard
back - Back space
black - Black ribbon shift
red - Red ribbon shift
CR/LF - Carriage return without line feed
A.M. - Alternate mode

CTSS Character Set

	0	1	2	3	4	5	6	7
0000	0	1	2	3	4	5	6	7
0010	8	9		=	'			
0020	+	A	B	C	D	E	F	G
0030	H	I		.	)	:		
0040	-	J	K	L	M	N	Ø	P
0050	Q	R	F.F.	\$	*	C.R.		null
0060	blank	/	S	T	U	V	W	X
0070	Y	Z	tab	,	(			
0100	(I)	(J)	(^)	(;)	(#)	(¬)	(@)	(L.F.) *
0110	(sngl)	(dbl)	(%)	(bell)	(!)	(WRU)	(hang)	(P-off)
0120	&	a	b	c	d	e	f	g
0130	h	i	(black)	(red)		back	(CR/LF)	"
0140	(_)	j	k	l	m	n	o	p
0150	q	r	(<)	([)	(prefix)	(>)	?	
0160		(L.K.)	s	t	u	v	w	x
0170	y	z	(V.T.)			(P-on)	(U.K.)	(A.M.)

NOTES:

1. Character codes in parentheses are discarded on input in 6-bit mode. In 12-bit mode these characters have (400)8 added to them, as a flag bit.
2. Character code 0137 (double quote) is the erase character in 6-bit mode.
3. Character code 0156 (question mark) is the kill character in 6-bit mode.
4. The codes 0017 (Interrupt), 0057 (Quit) and 0077 (Hang-up) on input are intercepted by the supervisor and are never sent through to the program.

1050 Character Set

Same as CTSS Character Set except as noted below:

	0	1	2	3	4	5	6	7
0000								
0010								
0020								
0030								
0040								
0050			ig					
0060								
0070								
0100	($\circ$)	($\pm$)	(α)			(%)		
0110			ig	ig	ig	ig		
0120								
0130							ig	
0140								
0150			(,)	(!)		(.)		
0160		ig						
0170			ig				ig	ig

1. This table assumes a "standard correspondence" ball. On machines with an "old-style correspondence" ball, code 0100 will print a degree sign, and code 0102 will print a cent sign.
2. Code 0152 will type an upper case comma on output; an upper case comma on input will be given code 0073.
3. Code 0155 will type an upper case period on output; an upper case period on input will be given code 0033.
4. Code 0017 (Interrupt) can be generated by one push of the "reset line" button.
5. Code 0057 (Quit) can be generated by two pushes of the "reset line" button.

Old Style, MIT-modified Model 35 Teletype

Same as CTSS Character Set, except as noted below:

	0	1	2	3	4	5	6	7
0000								
0010								
0020								
0030								
0040								
0050								
0060								
0070								
0100			(\)				(%)	ig *
0110	ig	ig	ig					
0120		A	B	C	D	E	F	G
0130	H	I	ig	ig		ig	ig	
0140	(←)	J	K	L	M	N	Ø	P *
0150	Q	R			ig			
0160			S	T	U	V	W	X
0170	Y	Z						

1. Some machines may type different graphics for any of all of the following:

Character Code	Normal Graphic	Alternate Graphic
0102	\	~
0104	#	~
0105	%	~
0106	@	v
0120	&	^

2. Machines using this character set are to be replaced with machines using the "Standard Model 35" character set as soon as practical.
3. Code 0017 (Interrupt) may be generated by one push of the "Break" button.
4. Code 0057 (Quit) may be generated by two pushes of the "Break" button.

Standard Model 35 Character Set

Same as CTSS Character Set except as noted below:

	0	1	2	3	4	5	6	7
0000								
0010								
0020								
0030								
0040								
0050								
0060								
0070								
0100	(\)				(%)			
0100	ig	ig	ig					
0120		A	B	C	D	E	F	G
0130	H	I	ig	ig		ig		
0140	(←)	J	K	L	M	N	Ø	P
0150	Q	R			ig			
0160			S	T	U	V	W	X
0170	Y	Z						

1. Some outside (i.e. not new-style MIT-modified) model 35's will not respond to code 0176 (Keyboard unlock).
2. On outside model 35's, code 0055 (Carriage return) will cause a Carriage Return and a Line Feed on output. The computer will type a line feed whenever a carriage return is detected on input.
3. On outside model 35's, the tabulate character (0072) prints as a back slash and will not cause tab motion of the carriage.
4. Code 0017 (Interrupt) may be generated by one push of the "Break" button.
5. Code 0057 (Quit) may be generated by two pushes of the "Break" button.

Telex Character Set

Same as CTSS Character Set except as noted below:

	0	1	2	3	4	5	6	7
0000								
0010	:							
0020	&							
0030	bell							
0040								
0050	ig #							
0060								
0070	;							
0100	ig	ig	ig			ig	ig	
0110	ig	ig			ig		ig	ig
0120		A	B	C	D	E	F	G
0130	H	I	ig	ig	ig	ig		
0140	ig	J	K	L	M	N	Ø	P
0150	Q	R	ig	ig	ig	ig		
0160		ig	S	T	U	V	W	X
0170	Y	Z	ig		ig	ig	ig	ig

NOTE:

1. Code 0115 (Who Are You) prints a Maltese Cross.
2. Either code 0035 or code 0113 will ring the Telex bell on output. On input, a bell produces code 0035. *
3. Either code 0072 or code 0103 will print a semicolon on output. On input, a semicolon produces code 0072.
4. Either code 0020 or code 0120 will print an ampersand on output. On input, an ampersand produces code 0020.
5. Either code 0054 or code 0104 will print a number sign on output. On input, a number sign produces code 0054.

Revised: 1/3/66

Identification

List contents of file directory

Purpose

To provide a command which lists the contents of a file directory, with numerous selectivity features if desired.

Description

LISTF enables the user to selectively list the contents of a file directory by permitting him to specify the

1. file directory
2. file names
3. authors
4. modes
5. range of dates last used
6. range of dates last modified
7. sorting process
8. output form

to be employed.

The user has the option to suppress the search for linked files or to search only for linked files.

Usage

A. Basic

The basic call - LISTF - will first produce a one-line summary of the number of nonlinked files and the number of records in the user's current directory. This is followed by a table of nonlinked files in the form

NAME1 NAME2 MODE NREC DATE(last used)

sorted according to the date last used with the most recent date first. This is followed by a one-line summary of the number of linked files and a table of linked files in the form

NAME1 NAME2 MODE(in user's directory) PROBN PROGN LNAME1 LNAME2

alphabetically sorted with respect to the primary file name, where the last four items refer to the "other end of the link."

B. Options

The selectivity features and their usage are described on the pages which follow.

Conventions

1. Arguments are divided into four classes.
 - a. meta-arguments (defined inductively from the Specifications Tables below)
 - b. modifiers (defined inductively from the Specifications Tables below)
 - c. file names - all arguments which cannot be identified as meta-arguments or modifiers
 - d. special characters
 - 1) carriage return
 - 2) '
 - 3) (and)
 - 4) *
2. A request is a string of arguments terminated by a single quotation mark (') or by a carriage return.
3. A call is the command LISTF followed by a string of requests and terminated by a carriage return.
4. The order of the arguments is unimportant, aside from the following considerations:
 - a. modifiers must immediately follow the meta-argument which they modify
 - b. when sorting by dates, the list will begin with the first date specified
 - c. two primary file names must be separated by a secondary name or by a meta-argument
5. Up to 19 arguments may be specified in one call to LISTF.
6. One interrupt level is set to enable the user to terminate the request being processed and begin the next. (WARNING: some output may be lost.)
7. If the user quits when he is listing linked files in a common file in the long form, his directory switching will probably not be restored. (This condition can, of course, be corrected by issuing a COMFIL 0 command.)

File Names

An asterisk (*) embedded in a file name specification refers to any and all characters in that position. A single * as a file name means any and all names.

EXAMPLES:

CTEST* means any name with "CTEST" as the first five characters, i.e., CTEST1, CTESTS, but not bCTEST, where "b" denotes blank.

TEST means any name with "TEST" as the 2-5 characters, i.e., CTEST1, bTESTS, but not bbTEST or TEST12.

** means any 1 or 2 character name.

If the secondary file name is omitted, * will be assumed.

If no file names are specified, * * will be assumed.

SPECIFICATIONS TABLES

SEARCH SPECIFICATIONS

<u>META-ARG.</u>	<u>MODIFIERS</u>	<u>ACTION</u>	<u>DEFAULT</u>	<u>COMMENTS</u>
(FILE)	NONE	ignores <u>linked</u> files		
(LINK)	NONE	ignores <u>non-linked</u> files		
			accepts both linked and nonlinked files	
(UFD)	name of a file linked to some file "U.F.D. (FILE)"	searches the files in the linked file directory	U.F.D. (FILE)	
(SYS)	NONE	searches the system files	current directory	
(CFIn)	NONE	searches the user's <u>n</u> th common file	current directory	
(AUTH)	authors number(s)	only accepts <u>non-linked</u> files written by user(s) specified	any and all authors	
(MODE)	1 or more descriptions, each comprising 1-4 of: O,T,S,R,W,V,L,P,* enclosed by parentheses	ignores files with modes not requested	any and all modes	e.g. (RP) means read-only, prot. (RP*) means at least read-only, prot. (R)b(P) means read-only or prot.
(USED)	up to 2 dates, each in the form MMDDYY, or '(OLD)', or '(OLD)', or '(NEW)'	only accepts <u>non-linked</u> files used between between dates, incl.	(NEW) (OLD)	(OLD) means earliest date (NEW) means (NEW) means present date
(MADE)	same as (USED)	only accepts <u>non-linked</u> files modified between dates, incl.	(NEW) (OLD)	same as (USED)

SORTING SPECIFICATIONS

<u>META-ARG.</u>	<u>MODIFIERS</u>	<u>ACTION</u>	<u>DEFAULT</u>	<u>COMMENTS</u>
(SDIR)	NONE	file directory order		
(SNA1)	NONE	alphabetic order NAME1		
(SNA2)	NONE	alphabetic order NAME2		
(SMOD)	NONE	sorted on(Octal) modes decreasing order		
(SREC)	NONE	number of records		<u>nonlinked</u> files *
(SUSE)	NONE	date last used		<u>nonlinked</u> files *
(SMAD)	NONE	date last modified		<u>nonlinked</u> files *
			(SUSE) for <u>nonlinked</u> files (SNA1) for linked files	
(REV)	NONE	reverses the order of sorting		

*
Listing of linked files will be suppressed in requests
containing these meta-arguments.

OUTPUT SPECIFICATIONS

<u>META-ARG.</u>	<u>MODIFIERS</u>	<u>ACTION</u>	<u>DEFAULT</u>	<u>COMMENTS</u>
(LSUM)	NONE	summary lines only		
(LNAM)	NONE	NAME1-NAME2 table		
(LONG)	NONE	normal form plus TIME and DATE (last modified), AUTH.,DEV.,LOCK for <u>non-linked</u> files; normal form plus MODE ("or"ed), DATE last used), DATE (last modified), AUTH.,NREC,DEV. for <u>linked</u> files		if (UFD) was re- quested, <u>linked</u> files will be printed in the normal form
			normal form- (see USAGE)	

Example 1

```

      (1) (2)  (3)   (4) (5) (6)  (7)
LISTF *  BSS  GAMMA* (AUTH) 1  2  (LONG)

(8)  (9)  (10) (11) (12)  (13)  (14) (15)
(MODE) (RP) (W*) (AUTH) 99999 (CFL2) (USED) 080165

(16) (17)  (18)  (19)
(SDIR) (MADE) 080165 080165

```

would produce a table in the long form, in the order
 of the file directory, of all files with the following
 properties

1. non-linked ^{(4),(11),(14),(17)}
2. secondary name "BSS" ^{(1),(2)} and/or primary
 name "GAMMA (any character)" ⁽³⁾
3. written by user no.s 1, 2 or 99999 ^{(5),(6),(12)}
4. in read-only, protected mode ⁽⁹⁾ or has write-only
 bit set ⁽¹⁰⁾
5. in common file 2 ⁽¹³⁾
6. last used on or before 9/1/65 ⁽¹⁵⁾
7. last modified on 8/1/65 ^{(18),(19)}

where the superscripts are, of course, for reference only.

Example 2

If present date is 12/31/65 and all files in the directory were last used between 1/31/65 and 12/31/65, inclusive, then the following requests would produce identical tables (consisting of all the non-linked files used from 1/31/65 to 12/31/65, inclusive, in the normal form beginning with the file last used).

1. (USED)
2. (USED) (NEW)
3. (USED) 123165
4. (USED) (NEW) (OLD)
5. (USED) (OLD) (REV)
6. (USED) 013165 123165 (REV)
7. (USED) 123165 013165
8. (USED) (NEW) 013165
9. (USED) 013165 (NEW) (REV)
10. (FILE)
11. (AUTH)
12. (MADE)
13. (REV)(REV)(USED)
14. (SUSE)

1. (USED)
2. (USED) (NEW)
3. (USED) 123165
4. (USED) (NEW) (OLD)
5. (USED) (OLD) (REV)
6. (USED) 013165 123165 (REV)
7. (USED) 123165 013165
8. (USED) (NEW) 013165
9. (USED) 013165 (NEW) (REV)
10. (FILE)
11. (AUTH)
12. (MADE)
13. (REV)(REV)(USED)
14. (SUSE)

"TIME-SHARING PRIMER"

INTRODUCTION

Beginnings are most difficult. This is far more true than trite in regard to the use of the Compatible Time-Sharing System (CTSS), which involves techniques that are liable to seem rather obscure even to experienced programmers. This document was designed, then, in order to facilitate the new CTSS user's transition from batch-processing orientations to a time-sharing orientation. It does not pretend to offer "sophisticated" information. Rather, it is intended to relieve the reader of the necessity of having to worry about ferreting out -- usually by word of mouth -- the basic operational information which is prerequisite to sophistication. So, leaving only the details of becoming an accredited user through administrative channels, and of turning on and dialing in his particular console (see Section AC.3) to the reader, we attempt to present here a "toehold", a guide (including an annotated "script") to the new CTSS user for his first time-sharing console session.

(The material herein is based upon "Some Introductory Notes on Time-Sharing Console Usage Techniques for the Summer Programming Course," which was written as a reference for students taking the one week Basic Programming and FAP Courses, offered annually by the M.I.T. Computation Center; as such, it may have rather too pedantic a cast -- though one hopes that over-simplification is more informative than over-complication.)

A QUICK LOOK AT TIME-SHARING

System:

Time-sharing is a system which allows a number of users to make use of a computer "at the same time" for independent tasks. The technique is possible because of the large mismatch between computer speeds and human reaction times. Although the computer is actually sharing its attention among all of its users, it can be made to appear to each user as if he had control of the machine in its entirety. The program which regulates the process of co-ordinating activities (the CTSS Supervisor -- or "the system") resides in a separate bank of core memory, and actually causes the various users' programs to be brought into a second bank of memory from other storage devices.

Interaction:

Each user really has physical control only over some remote input-output terminal, usually a typewriter-like

CB1 81 box 2

Feb 14, 1966

device ("console"). He issues basic instructions, called "commands", to the system by typing the name of the command and the arguments associated with it; the system will then bring in the program which is associated with (i.e., "which performs") the command and cause it to be executed. In general, the user types in lower case and the system's responses are in upper case (except on teletype devices, which operate only in upper case). When the system receives a command it acknowledges receipt by typing out on the user's console a line comprising the letter "W" (for Wait) followed by a five-digit number expressing the time of day. When the command has finished working, the system informs the user of this fact by typing a line comprising the letter "R" (for Ready) followed by two numbers separated by a plus sign, the first number expressing the number of seconds expended in executing the command and the second number expressing the number of seconds expended "swapping" the program(s) involved in and out of core.

The user is said to be at "command level" after receiving an "R"(Ready) line. When at command level, he may issue any system command desired. During the execution of a program, however, commands are not accepted; in particular, as commands themselves are programs they can not be over-ridden by the typing of new commands. In order to return to command level before the executing program has finished, then, the user must give a "quit signal" to the system. This quit signal is two pushes of the console's "break" button. The ability to quit is quite useful, especially when, for example, a user's program misbehaves or a command has furnished enough information for one's purposes but would continue to operate "longer" if not interrupted.

Files:

Most often, the arguments of commands are the names of "files" where a file is broadly defined as a logical set of information. A file may contain ("the information may represent") a source program, an object program, a set of data, text, lists, or almost anything definable by the user which is expressible in the symbols available. These files may be input from CTSS consoles or from punched cards (see Section AE.1) and are normally stored on the computer's magnetic disk storage devices; however, their actual location is of no importance to the programmer since he always refers to files by name. The system itself provides for references to actual locations internally and maintains a separate "file directory" for each user so that no conflicts arise in the assigning of names.

A master file directory (M.F.D.) is maintained, containing information about the location and contents of the several user file directories (U.F.D.). Each U.F.D.

contains information about the location and contents of the various files which the user has created. The U.F.D. is associated with a problem number and a programmer number. Also associated with certain problem numbers are "common files" -- file directories which contain files of common interest and are directly accessible to all users on the problem number.

Certain of the common files associated with the system programmers' problem number (M1416) contain information of general utility and are accessible to all users. (See Section AD for further information about files.)

Each file is required to have two names, a "primary" name and a "secondary" name, each of which consists of six or fewer characters. The primary name is almost always arbitrary and should have some mnemonic importance. The secondary name may or may not be arbitrary, depending on the contents of the file and the way in which they are to be used. For example, a file containing a MAD (Michigan Algorithm Decoder) source program may have the arbitrary primary name PROG1, but must have the secondary (class) name "MAD". Object program files have the secondary name "BSS" (Binary Symbolic Subroutine).

Applications:

Learning to use CTSS is similar to learning to play the guitar. Knowledge of a few basic chords enables the novice musician to play a rather large number of songs; knowledge of a few basic commands enables the novice CTSS user to write and execute an arbitrarily large number of programs in a rather large number of programming languages (Section AH.2). Beyond this basic area of application (which is the only one dealt with in detail here), however, are at least two other large areas of special application. In the first place, there exists a large number of special-purpose commands for such purposes as file manipulation, debugging, documentation, and interactive problem-solving (Section AH). In the second place, user programs may avail themselves of a wealth of library subroutines, both batch-processing and time-sharing in nature (Section AG). By taking advantage of these additional tools, the CTSS user may expand his repertoire of applications as necessary, and probably more rapidly than the guitar player expands his repertoire of songs.

Reference:

Further general information of interest may be found in Sections AA.0 and AA.1. Information about the use of the manual may be found in Section AB.

OVERVIEW OF BASIC CONSOLE TECHNIQUES USED IN PROGRAM CREATION

1. Typing errors in command lines and in input lines may be corrected by typing a question mark (?) to cause the system to ignore ("kill") the entire line thus far, or by typing one or more quotation marks (") to cause the system to ignore ("erase") one or more immediately preceding characters.
2. A console session is begun (after turning on and dialing in the console) by issuing the LOGIN command, identifying the user to the system and establishing that a line to the computer is available.
3. Source programs will be entered and modified or corrected using the text editing command EDL. (Other available editing commands are covered in Section AH.3.)
4. Compilation will be accomplished by the MAD command in this document -- although other compilers and assemblers are available in CTSS (Section AH.2.)
5. Once a program has been successfully compiled (or assembled), execution is effected by the LOADGO command--again for purposes of this documentation; loading of programs is covered generally in Section AH.7.01.
6. When a program has been satisfactorily run, it may be removed from the user's file directory by use of the DELETE command. Files not explicitly deleted will be left alone and will still reside in the disc storage units.
7. At the end of a console session, the LOGOUT command is given to inform the system that the user's line to the computer is free to accomodate someone else.

DESCRIPTION AND DISCUSSION OF COMMANDS

The LOGIN Command:

After the console has been turned on and dialed in, type a line of the following general form:

"login probno name",

where probno is an argument of the LOGIN command specifying the user's assigned problem number, and the second argument

is the user's last name. Commands and arguments must be separated by at least one blank ("space"). The system will respond with a w(ait) line, and then will type out "PASSWORD". At this point, the user must type his assigned private password, during which time the console's printing will be suppressed. Provided a line is available -- and the user has been allotted time and disk storage records on the system -- a message acknowledging the fact that the user has been "logged in" will follow. (Further details may be found in Section AH.1.01).

When there is no line available, the system will cause the console to be "hung up" (disconnect at the system's end of the connection), and the user should try to log in at a later time. If, on the other hand, no response is typed after the login command was given, CTSS is not in operation; information about when it is expected to be back in operation may be gotten from Ext. 5948 (MAC machine) or Ext. 4121 (Computation Center machine). Occasionally, the system will not recognize "login" as a command; this means that the name of the login command has been temporarily altered so that the system programming staff can hold a test session.

The EDL Command for Input:

1. EDL is a CTSS command which is used for input and for "context editing" of files. We will take advantage of its input facility, to create the files which will later be edited. "Context editing" requires the unique specification and location of a line in terms of its contents by means of appropriate requests ("subcommands" of EDL) before the line can be edited. (This rather obscurely-stated point should be made clear by discussion below -- EDL for Editing, point 5 -- and by the Appendix). Requests to EDL may be abbreviated by their first letter, with the exception of the request "file" (see point 8), although the full request name may also be used; the abbreviated forms will be used herein.
2. A more complete description of the EDL command may be found in Sections AH.3.07 and AH.9.01.
3. To begin input: type, e.g., "ed1 abc123 madCR" or, in general, "ed1 name1 name2CR", where CR indicates Carriage Return.
4. Response: FILE ABC123 MAD NOT FOUND.
Input
"Input" is one mode of the EDL command; "Edit", the command's other mode, is discussed below.

5. For all lines which do not begin with statement labels, strike the "Tab" key, then type the line. For lines with labels: type the label, then strike Tab and type the rest of the line. For MAD continuation card indicators: tab, backspace, indicator, line. When finished with a line, strike Carriage Return (CR).

N.B. Wherever CR is indicated, strike the appropriate key on the console; do not type the letters "CR".

6. To deal with typing errors while still working on the line in which they occur: The quotation mark (") serves as an erase character and causes the ignoring of the immediately preceding character; more than one erase character may be used (e.g., XXY"YZ will be treated as XYZ by the computer). To kill the entire current input line, strike the question mark (?). N.B. This also deletes tabs, e.g., "/tab/x=a+b?y/tab/x=a1*b" causes "y/tab/x=a1*b" to be treated as the input line. A kill character cannot be erased.

7. For typing errors discovered in prior input lines: follow the procedures discussed below under the EDL command for editing (beginning with point 4).

8. To file a program: strike an extra CR (i.e., CR after last line plus CR for an "empty" line); this action causes entry to the Edit mode in which the request "file" may be used. Response will be a system R(eady) line, and the user will be at "command level" again -- which he was not while using EDL. (It is important to distinguish between general system commands, on the one hand, and requests to a specific command, on the other.) A file named, e.g., "abc123 mad" will have been established in the user's file directory.

N.B. EDL WILL ACCEPT REQUESTS IN THE EDIT MODE ONLY; in the Input mode, all material typed is treated as input.

9. To verify input (optional): type (general form) "print name1 name2". The PRINT command will cause the file to be typed back on the console.

The MAD Command for Compilation:

1. To cause the MAD (or the appropriate language's) compiler to operate on a program: the command is the name of the language and the argument is the primary name of the file; e.g., "mad abc123". The secondary name of the source file must be "MAD".
2. Response from successful attempt: A line beginning "LENGTH," followed by various other information. A file named , e.g., "abc123 bss" will have been created.
3. Error messages: These indicate "syntactic" mistakes; the source program file must be appropriately corrected.
4. Further details may be found in Section AH.2.10.

The EDL Command for Editing:
(CR Indicates Strike Carriage Return)

The following is excerpted from Section AH.9.01:

Editing is done line by line. We may envision a pointer which at the beginning of editing is above the first line of the file. This pointer is moved down to different lines by some requests, while other requests specify some action to be done to the line next to the pointer. All requests except FILE may be abbreviated by giving only the first letter. Illegal or misspelled requests will be commented upon and ignored.

The Appendix and the discussion below should clarify the importance of the "pointer". Requests which take arguments must be separated from the arguments by a space.

1. Type "ed1 name1 name2CR" (general form).
2. Response should be "Edit".
3. The EDL command will type back lines ("verify" them) after certain requests. The requests which will cause verification are "locate" and "change" (discussed below); wait for the response before issuing another request when one of these two has been given.
4. Type "tCR" ("t" is the abbreviation for "top"). (This is not strictly necessary for beginning to edit, but is required when the Edit mode has been entered from the Input mode, or when the pointer

must be moved "upwards".) The "pointer" is positioned "above" the first line of the file. Note that "top" is the only request to EDL which moves the pointer "upwards".

5. To position the pointer to a particular line, use "l" (for "locate"). The argument of this request (typed after a space which must follow the "l") is a string of characters which uniquely specifies a line amongst the lines "below" the pointer. The pointer will be moved to the line which contains the first occurrence of the string. E.g., if the "top" request had just been issued and the first two lines of a file were

A = B+C

D = A+X

the request "l aCR" would position the pointer at the first line, but "l a+CR" would have positioned it at the second line. (Note also that in the latter case the first line is then "above" the pointer, and if it is to be operated upon, the "t" request - "l" request sequence must be given again.)

6. To replace an entire line, the request is "r" (for "retype"). The argument (typed after a space which must follow the "r") is the entire new line itself (with appropriate tabs and terminal CR, as in input). This request does not move the pointer.
7. To change a portion of a line, the request is "c" (for "change"). The argument (space as usual) is rather complex: Begin with an arbitrary character which does not appear in either the original string of characters to be changed or the new string ("q" is frequently useful); this character serves as a delimiter of the two strings. Between the delimiters, type the old and the new character strings, in that order. The first occurrence of the old string will be altered. For example, "c qabcqxyzqCR" will cause "abc" to be replaced by "xyz", and if the original line were "abcabc" the resulting line would be "xyzabc". Blanks within the strings are significant: "a bc" is not the same as "abc". (This request does not move the pointer.) "Global" changes are possible, but will not be dealt with here.
8. To insert one line after the line currently pointed at, type "i", followed by a space, followed by the line to be inserted. To insert

several lines, change mode from Edit to Input by giving an "extra" CR, or by typing "iCR". The response will be "Input". Type the line or lines, with appropriate tabs. When done inserting, return to Edit mode by giving an extra CR.

9. To delete a line or lines: position the pointer (with the "I" request) to the first line to be deleted, then type "d" (for "delete") followed by CR if only this one line is to be deleted, or by a space and a number (expressing the number of consecutive lines to be deleted) if more than one, then CR. (This request leaves the pointer positioned at the last line deleted.)
10. To move the pointer "downward" one or more lines, the request is "n" (for "next"); it takes a numerical argument, in the same fashion as "d".
11. To re-file under the original file name, simply type "fileCR" (from the Edit mode). This process replaces the older version with the edited version.
12. To file under a new file name, type "file xxxxxxCR" where xxxxxx represents the new primary name. This process preserves the older version, in the event that a comparison of both versions is desired for some reason (e.g., to determine which of two methods takes longer). Secondary names may not be changed when filing.

The LOADGO Command for Execution:

1. After a successful compilation or assembly (no syntactical errors) has been achieved, the command "loadgo name1" will cause the object program ("name1 bss") to be loaded and executed. Library search occurs during the loading process.
2. Shortly after the customary w(ait) response, the word "EXECUTION" will be typed by the system. This will be followed by the program's results, if all has gone well with the program. Then, provided there were no execution errors, an end-of-run message and a system R(eady) line will be typed out.
3. Further details may be found in Section AH.7.01.

Program Logic "Debugging":

1. Wrong results imply errors in program logic. (See

CC Memo 182 for a list of common programming errors.)

2. When discovered, the errors can be corrected in the source file (name1 mad, e.g.,) with the EDL command.
3. After editing, the program must be recompiled with the appropriate language command (MAD).
4. The new program is executed with the LOADGO command.
5. If the results are still wrong, back to 1....

Housekeeping:

When a program is no longer desired, all files relating to it can be removed from the disk by typing (general form) "delete name1 *CR". The asterisk indicates to the DELETE command that it is to operate on all files with primary name "name1". Of course, individual files may be dealt with by "delete name1 name2". (Further details may be found in Section AH.6.03).

The LOGOUT Command:

At the end of a console session, give the command "logout". The system will respond with the present time, the date, and the total time used (in minutes). (Further details may be found in Section AH.1.02.)

APPENDIX: CONSOLE FAMILIARIZATION SESSION -- AN ANNOTATED SCRIPTIntroduction:

The program created in this script is deliberately simple-minded, so as not to distract from the basic point at issue -- console usage. (For demonstration purposes, some of the errors introduced are unique to the MAD language, but should be reasonably clear to the reader even if he is not familiar with MAD). The program is intended merely to compute and output the square root of the sum, and the product, of two numbers input from the console. (Data can, of course, be input to the program from files as well as from the console. Indeed, batch processing tape techniques are simulated on CTSS -see Section AG.5 -- and numerous subroutines are provided for direct disk file I/O -- see Section AG.2.)

Instructions:

1. Type the lines appearing in lower-case letters and wait for the system responses if a line in upper-case occurs next in the "script".
2. Hit Carriage Return at the end of each lower-case line.
3. Circled numbers to the left of the page refer to the Notes, which follow the "script".
4. Long-hand insertions are typing instructions (usually involving the Tab key) e.g., *Tab* .
5. The numbers in W(ait) and R(eady) lines are fictitious; expect different ones.
6. Before issuing the DELETE command, the LISTF command may be used to get a listing of the contents of your file directory (Section AH.5.01), and TTPEEK may be used to get a table of your time and track usage for the current month (Section AH.1.04). Neither command requires arguments.

Script:

login m1416 padlipsky

W 1315.1

PASSWORD

PARTY LINE BUSY, STANDBY LINE HAS BEEN ASSIGNED

M1416 3711 LOGGED IN 1/10/66 1315.6 from 2000K

CTSS BEING USED IS DMN2C5

LAST LOGOUT WAS 1/6/66 1756.0
R 6.783+750

edl simple mad

W 1316.4

FILE SIMPLE MAD NOT FOUND.

Input

normal mode is integer

floating point a

Tab print comment\$numbers,pleeuhz""ase\$

Tab read data

Tab a=sqrt(b+c)

Tab d=bc

end of? *Tab* end of program

Edit

t

l mode

NORMAL MODE IS INTEGER

r *Tab* normal mode is integer

n

r *Tab* floating point a,d

l a

PRINT COMMENT\$NUMBERS,PLEASE\$

l a=

A=SQRT(B+C)

c qtqt.q

A=SQRT.(B+C)

l d=

D=BC

i

Input

Tab print results a,d

Tab execute exit.

Edit

file

R 5.833+4.250

print simple mad
W 1321.3

SIMPLE MAD 01/10 1321.4

NORMAL MODE IS INTEGER
FLOATING POINT A,D
PRINT COMMENT\$NUMBERS,PLEASE\$
READ DATA
A=SQRT.(B+C)

D=BC
PRINT RESULTS A,D
EXECUTE EXIT.
END OF PROGRAM

R .616+416

mad simple

W 1321.9

THE FOLLOWING NAMES HAVE OCCURRED ONLY ONCE IN THIS PROGRAM.
COMPILATION WILL CONTINUE.

BC

B

C

LENGTH 00072. TV SIZE 00006. ENTRY 00016
R 2.766+.533

edl simple mad

W 1322.8

Edit

l bc

D=BC

c qbqb*q

D=B*C

file

R 3.516+1.450

mad simple

W 1323.7

LENGTH 00071. TV SIZE 00006. ENTRY 00015

R 2.216+.750

loadgo simple

W 1324.1

EXECUTION.

NUMBERS, PLEASE

b=7,c=2*

A = 2.707999E 26,
EXIT CALLED. PM MAY BE TAKEN.
R 6.166.1.050

D = 14.000000

edl simple mad

W 1325.5

Edit

l mode

NORMAL MODE IS INTEGER

d

l read

READ DATA

i *Tab*

l exit

whenever (b+c).1.0., transfer to tag

EXECUTE EXIT.

c qqtag2q
TAG2 EXECUTE EXIT.
i
Input
tag ^{Tab} print comment\$negative arguments\$
~~Tab~~ transfer to tag2

14

Edit
file
R 2.950+3.150

mad simple
W 1523.1
Length 00107. TV SIZE 00006. ENTRY 00020
R 2.966+.900

loadgo simple
W 1523.7
EXECUTION.
NUMBERS, PLEASE
b=7.,c=2.*

15

16

A = 3.000000, D = 14.000000
EXIT CALLED. PM MAY BE TAKEN.
R 6.566+1.083

loadgo simple
W 1524.7
EXECUTION.
NUMBERS, PLEASE
b=-7.,c=2.*

17

NEGATIVE ARGUMENT
EXIT CALLED. PM MAY BE TAKEN.
R 6.216+.816

delete simple *
W 1526.1
R 1.716+366

18

logut
W 1528.2
LOGUT NOT FOUND.
READY.

19

logout m1416""""""
W 1528.4
M1416 3711 LOGGED OUT 1/10 /66 1536.3 FROM 20000K
TOTAL TIME USED= 00.7 MIN.

Notes:

1. We decide pleeeuhz isn't funny and use four erase characters.
2. The "empty line" takes us to Edit mode.
3. The line should have been tab'ed originally.
4. Same as 3, and we realize we want both answers floating.
5. These two locates demonstrate "context editing".
6. We remember that MAD subroutine calls require periods.
7. This insertion allows us to see the answers after execution and terminates the program in standard fashion.
8. The "empty line" again.
9. Verifying the typing.
10. The error message reminded us that we meant BC to be a product, not a name.
11. Whoops! A is 'way wrong. We have a bug.
12. We remember that the square root routine expects floating point arguments, and take the easiest route of getting them to be floating -- deleting the integer mode declaration.
13. We remember that the square root routine also expects positive arguments.
14. Still another "empty line".
15. N.B. the decimal points.
16. Success.
17. And success again.
18. The misspelled command is not findable.
19. Erase characters apply in command lines too.

(END)

Revised: 2/14/66

Identification

Mail command
MAIL

Purpose

To place a file containing a message to another user in his file directory, whether he is logged in or not.

Usage

```
MAIL  NAME1  NAME2  PROBi  PROG1  ... -PROBn- -PROGn-  
MAIL  NAME1  NAME2  '(LIST)' LNAME1  LNAME2
```

*

NAME1 NAME2 is the name of the file to be mailed. It must be line-marked, and no more than 1 record in length.

PROBi PROG1 are the users to which mail will be sent.

(LIST) If the '(LIST)' option is given, the file LNAME1 LNAME2 will be used as a "mailing list", and mail will be sent to all PROBi PROG1 pairs in the file. The file may be card-image or line-marked; its format is free, except that items must be separated by spaces.

*

Mail will be placed in a file named MAIL BOX in the records of user PROBi PROG1. If the file already exists, it will be appended to. Each piece of mail is prefaced with a message of the form "FROM USRPB USRPG DATE TIME" where USRPB is the sender's problem number, USRPG is the sender's programmer number, and DATE and TIME have the usual meanings. (To ascertain whether he has received mail, the user should periodically - daily, perhaps - issue the command 'PRINT MAIL BOX'. Because of the appending feature of the MAILing process, the command 'DELETE MAIL BOX' should be issued after a message has been PRINTed, to avoid having to run through previous messages to get to the latest one.)

*

Any PROBi or PROG1 may be '*', meaning "all"; the command will search the MFD and send mail to all users (but not to common files) satisfying the criterion. However, '* *' will not cause mail to be sent to all users.

*

Typing the command 'MAIL' without arguments is equivalent to asking for instructions on how the command is to be used.

*

To avoid getting mail, one may place in his tracks a file of name MAIL BOX with PRIVATE mode.

If an addressee is over his record quota, 'MAIL BOX' will be *
written in temporary mode.

Restriction

If the receiver's MAIL BOX is PRIVATE, PROTECTED or
READ-ONLY, mail cannot be delivered.

(END)

Identification.

Binary file editing program (EDB)
J. H. Saltzer, x6039 2/14/66

Purpose.

EDB is a reincarnation of the TYPSET and EDL editing commands for use with arbitrary binary files.

Usage.

RESUME EDB name1 name2

will allow creation or editing of the file "name1 name2". The editing conventions of EDB are identical to those of EDL and TYPSET, as described in section AH.9.01. Each 36 bit word of the file being edited is a distinct line in the sense of the TYPSET description, and is represented for editing purposes as a 12-digit octal number. Care should be taken to insure that after input or editing, there are exactly 12 octal digits in a line. If there are more than 12 octal digits, the last 12 will be used; if fewer, leading zeros will be appended. Non-octal characters will be converted to octal by truncation of the high order bits.

(END)

Identification

Fixed File Names

Introduction

Unexpected file names appear in a user's file directory from time to time. The following is a partial annotated list of files generated:

- 1) by CTSS in performing system duties;
- 2) by one of the commands which makes a new file as part of its execution process; or
- 3) by another CTSS user.

Note, and be warned, that catastrophic conflicts will arise if several users are performing a command which generates a fixed file name at the same time in the same file directory (usually a common file). The obvious way to avoid such conflicts is to avoid performing such commands while attached to any directory other than one's "home directory".

Files With Both Names Fixed

(COMBI NFILE) AH.6.01:

This name is given to the intermediate file employed by the COMBIN command. It may be deleted if found.

(INPUT FILE) AH.9.01,
(INPT1 FILE) AH.9.01:

These two names are used for intermediate files by TYPSET, ED, and EDL. Following a quit sequence (or an automatic LOGOUT) either one of these files may be found. It may be renamed and used as a source file (in the automatic LOGOUT case, editing may, of course, be continued when the LOGOUT SAVED file is resumed). When invoked, the editing commands will announce the presence of one of the intermediate files (if one is present); the user must either type 'yes' to the question about deleting it, or type 'no' and then RENAME it. The commands will not proceed unless the intermediate file is disposed of, one way or another.

LOGOUT SAVED AH.1.02:

At any time an automatic LOGOUT may be initiated by the system. If the user is not in dead status, this SAVED file will be created. The file may be resumed at a later time.

CB1 G1 5x2

May 31, 1966

MAIL BOX AH.9.05:

This file is created (or appended to) when a user gives the command MAIL with the problem number and program number of the addressee's file directory. When the recipient subsequently logs in the following message will appear on his console:

YOU HAVE MAILBOX.

(MOVIE TABLE) AH.7.01, AJ.8.01:

The MOVIE TABLE is created by the standard loaders. It is a temporary mode file and represents a map of the programs loaded.

OUTPUT REQUEST AH.6.06:

The REQUEST command for bulk I/O creates or appends to a file in the user's disk storage. When the file has been processed by the disk editor, it is set to temporary mode.

PERMIT FILE AH.3.05:

The PERMIT command establishes a line-marked file of private protected mode in the user's directory; PERMIT FILE contains information used in the linking process.

URGENT MAIL AH.1.01,
URGENT POST AH.1.01:

DAEMON can create this file in a user's directory so that his subsequent LOGIN will remind him TO PRINT the new file in order to get a message from the system. The alert message printed on his console is:

YOU HAVE URGENT MAIL
or
YOU HAVE URGENT POST.

Files With Fixed Second Names

NAME1 BCD AH.2.07, AH.2.10, AH.2.11:

A file of secondary name 'BCD' is produced by several of the language processors on request. Such files contain assembly/compilation listings; they are generated in response to the argument '(LIST)' in the language processor command. ('BCD' is also the required secondary name for RUNCOM files; see AH.10.01.)

NAME1 BSS e.g., AH.2.07, AH.2.10,
AH.2.11:

A 'BSS' file contains an object program, produced by one of the language processors. 'BSS' is a 7094 term, documented elsewhere.

NAME1 (DUMP) AJ.8.03:

For details, see the DUMPER SAVED write-up.

NAME1 (MEMO) AH.9.01, AJ.6.01:

TYPSET creates a file with the secondary name '(MEMO)'. RUNOFF expects a file with the secondary name '(MEMO)'.

NAME1 SAVED AH.3.03:

'SAVED' files contain machine conditions and core-images, for subsequent execution. For details, see the SAVE write-up.

NAME1 SQZBSS AJ.4.04:

'SQZBSS' files contain compressed-form BSS "decks". For details, see the write-up on PADBSS SAVED and SQZBSS SAVED.

NAME1 SYMTAB AH.2.10, AH.2.11:

This is an optional file containing a symbol table, produced by the MAD (and, of course, MADTRN) language processor in response to the '(SYMB)' argument.

NAME1 SYMTB AH.2.07:

This is an automatically-generated file containing a symbol table, produced by the FAP language processor.

Files With Special or Fixed First Names

PROBNO PROGNO AH.4.01:

This is an intermediate file used by the ARCHIV command (where PROBNO, PROGNO are a user's problem number and programmer number). For details, see the ARCHIV write-up.

.TAPE. 3 AG.5.01:

The .PUNCH, .PNCHL and (SCH) subroutines create or append to a pseudo-tape line-marked file named .TAPE.3.

.TAPE. n AG.5.01:

The .TAPWR, (STH) and (STHM) subroutines create or append to a pseudo-tape line-marked file named .TAPE. n, where n is specified in the calling program.

...xxx SAVED AG.8.02:

This is an intermediate file used in chaining commands. For details, see the SCHAIN write-up.

...00n SAVED AH.3.04:

This is an intermediate file used in chaining commands. For details, see the RUNCOM write-up.

(END)